Aufzählungstyp - Enum

- Enum

Aufzählungstyp - Bedeutung

Bedeutung

- Oft Datentypen mit bestimmten Werten gebraucht
- Weder Zahlen noch Wahrheitswerte
- Beispiele
  - Farben: rot, grün, gelb
  - Wochentage: Mo, Di, Mi, Do, Fr, Sa, So
  - Spielerpositionen: Torwart, Kreisläufer, Rückraum, Außen
- Codierung über Zahlen möglich, aber irreführend
- Aufzählungstyp erlaubt Typdefinition mit begrenzter Wertemenge
- Synonyme: *Enumeration*

Aufzählungstyp - Syntax

Syntax

- **enum** *enumtype* {*enumelement* {, *enumelement*}*}
- Aufzählungstyp benannt durch *enumtype*

Konventionen

- *enumtype* wie Klassennamen
- *enumelement* groß geschriebenes, englisches Substantiv

Beispiele

- `enum Color {Red, Green, Blue, Yellow}`
- `enum Day {Mon, Tue, Wed, Thu, Fri, Sat, Sun}`

Eigenschaften

- Einzelne Werte innerhalb des Aufzählungstyps eindeutig!
- Gleichnamige Werte in verschiedenen Aufzählungstypen inkompatibel
- Zugriff: `Day.Mon`

Aufzählungstyp - Verwendung

Gegeben

- `enum Color {Red, Green, Blue, Yellow}`

Verwendung

- Gleichberechtigt zu anderen Typen in Java
- Definition einer Variablen
  - `Color c;`
- Zugriff auf Literal eines Aufzählungstyps
  - `Color.Red;`
- Zuweisen eines Wertes
  - `c = Color.Red;`
- Vergleichen von Werten / Objekten eines Aufzählungstyps
  - `if (c == Color.Red) …;`
  - Vergleich mit `==` ausreichend, `equals()` nicht erforderlich!

Aufzählungstyp – Vergleichbare Klassendefinition

Eigenschaften

- Aufzählungstyp ist spezielle Art von Klasse
- Also auch ein Referenztyp
- Aufzählungswerte sind finale statische Datenelemente
- Zur Laufzeit kein neuer Aufzählungswert erzeugbar!

Aufzählungstyp

```
enum Color {Red, Green, Blue, Yellow}
```

Vergleichbare Klassendefinition

```
class Color {
      final static Color Red = new Color();
      final static Color Green = new Color();
      final static Color Blue = new Color();
      final static Color Yellow = new Color();
}
```


Aufzählungstyp – Vordefinierte Methoden

Bestimmte Methoden vordefiniert für jeden Aufzählungstyp E

- `boolean equals(Object x)`
  - Vergleicht aktuellen Aufzählungswert mit `x`
  - Liefert `true`, wenn beide gleich sind, sonst `false`
- `int compareTo (E x)`
  - Vergleicht aktuellen Aufzählungswert mit x bzgl. Reihenfolge der Definition
- `int ordinal()`
  - Liefert den Index des aktuellen Aufzählungswertes bzgl. der Definitionsreihenfolge
  - Erster Wert hat Index 0
- Weitere vordefinierte Methoden verfügbar

Aufzählungstyp – Eigene Methoden

Definition eigener Methoden

```
enum Day {
  Mon, Tue, Wed, Thu, Fri, Sat, Sun;

  boolean isWeekend(){
    return this==Sat || this==Sun;
  }
}
```

Aufruf mit Aufzählungswert als Zielobjekt

```
Day today = …;
if(today.isWeekend()) …
```


Aufzählungstyp – Eigene Methoden

Definition von Datenelementen in Aufzählungstyp möglich

Beispiel: Datenelement zum Kennzeichnen von Wochenendtagen

- Test über Bedingung nicht mehr notwendig, nutzen von `weekend`
- Initialisieren des Datenelementes in einem Konstruktor
- Argumente für Konstruktor in Definition mit aufgezählt

```
enum Day {
  Mon(false), Tue(false), Wed(false), Thu(false),
  Fri (false), Sat(true), Sun(true);
  private final boolean weekend;
  Day (boolean w) { weekend = w; }
  boolean isWeekend(){ return weekend; }
}
```

Compiler verhindert Erzeugen neuer Aufzählungswerte (trotz Konstruktor)